IN THE CLAIMS

Please amend Claims 2, 42, 44-45, 47-50, 55-58, 60-63, 65, 67, 74-75, 79, 81, 85-86, 92, 98-99, 105, 107, 111, 112, 121-123, 127-135 as follows:

1. (Cancelled)

2. (Currently amended) For use in a system involving an Embedded-DRAM processor, a method for intelligent caching comprising the steps of:

splitting an architecture into first and second portions, said first portion comprising a set of functional units and a set of architectural registers exercised thereby, said second portion comprising at least one functional unit capable of moving data between a main memory implemented as one or more banks of DRAM without a caching system that employs cache hits and cache misses, and said set of architectural registers; and

splitting a single program into first and second concurrently executing portions which each concurrently execute distinct subsets of parallely dispatched instructions from one or more instruction streams, said first portion of said program executed on said first portion of the architecture, said second portion of said program executed on said second portion of said architecture;

wherein said second portion of said architecture is operative to prefetch data from said main memory into said architectural registers prior to being processed by said first portion of said architecture, and wherein said second portion of said architecture is operative to move results produced by said first portion of said architecture into main memory after they are produced by said first portion of said architecture; and

prior to when said first portion of said architecture executes a conditional branch instruction having a condition, said second portion of said architecture prefetches first and second data sets from memory into said architectural registers, said first data set being needed for use as instruction operands when said condition evaluates to true, said second data set being needed for use as instruction operands when said condition evaluates to false.

3. – 41. (Cancelled)

42. (Currently amended) The method of Claim 2, wherein speculative prefetching of data is performed from said main memory so that the first portion of said program need not wait

for the first or the second data set to be fetched from said main memory, irrespective of the outcome of the conditional instruction.

43. (Cancelled)

44. (Currently amended) The method of Claim 2, wherein the second portion of said architecture generates a row precharge instruction to precharge a DRAM row of a bank of the one or more banks of DRAM to cause data to be ready prior to issuing a read command.

45. (Currently amended) The method of Claim 2, wherein the set of architectural registers comprises a register file comprising a plurality of registers and having a parallel access port operative to load or store, under control of the second portion of said program, contents of said register file in a single DRAM access cycle from or to a DRAM row of a bank of the one or more banks of DRAM.

46. (Previously presented) The method of Claim 45, wherein the load operation is performed with a mask to allow certain of the contents of selected registers of the register file not to be modified by the load operation.

47. (Currently amended) The method of Claim 45, wherein said register file further comprises at least a second access port operative to transfer data between one or more of the functional units of the first portion of said architecture.

48. (Currently amended) The method of Claim 45, wherein the first portion and the second portion of said architecture cooperate to execute a DRAM row selected by a row-address register, and said register file further comprises at least a second access port operative to transfer data between one or more of the functional units of the first portion of said architecture.

49. (Currently amended) The method of Claim 45, wherein the register file can be placed into an inactive state where the register file does not appear in ~~the~~ a register space of the functional units of the first portion of said architecture.

50. (Currently amended) The method of Claim 45, wherein when the register file is placed into ~~the~~ an inactive state, the second portion of the architecture is enabled to cause a parallel load or store operation to occur between the parallel access port and a row of DRAM in a bank of the one or more banks of DRAM.

51. (Previously presented) The method of Claim 2, wherein the first and second portions of said architecture cooperatively execute instructions to process image data for display.

52. (Previously presented) The method of Claim 2, wherein the first and second portions of said architecture cooperatively execute instructions to process video data for display.

53. (Previously presented) The method of Claim 2, wherein the first and second portions of said architecture cooperatively execute instructions to perform digital filtering operations.

54. (Previously presented) The method of Claim 2, wherein the first and second portions of said architecture cooperatively execute instructions to execute a video decoder algorithm.

55. (Currently amended) For use in a system involving an Embedded-DRAM processor, a method for intelligent caching comprising:

providing an architecture split into first and second portions, said first portion comprising a set of functional units and a set of architectural registers exercised thereby, said second portion comprising at least one functional unit capable of moving data between a main memory implemented as one or more banks of DRAM, and said set of architectural registers, wherein the second portion accesses main memory without a caching system that employs cache hits and cache misses; and

providing a single program split into first and second concurrently executing portions which each concurrently execute distinct subsets of parallely dispatched instructions from one or more instruction streams, said first portion of said program executed on said first portion of the architecture, said second portion of said program executed on said second portion of said architecture;

wherein said second portion of said architecture is operative to prefetch data from said main memory into one or more of said architectural registers prior to being processed by said first portion of said architecture, and wherein said second portion of said architecture is operative to move results produced by said first portion of said architecture into main memory after they are produced by said first portion of said architecture; and

prior to when said first portion of said architecture executes a conditional branch instruction <u>having a condition</u>, said second portion of said architecture prefetches first and second data sets from said main memory into said architectural registers, said first data set being needed for use as instruction operands when said condition evaluates to true, said second data set being needed for use as instruction operands when said condition evaluates to false.

56. (Currently amended) The method of Claim 55, wherein speculative prefetching of data is performed from said main memory so that the first portion of said program need not wait for the first or the second data set to be fetched from said main memory, irrespective of the outcome of the conditional instruction.

57. (Currently amended) The method of Claim 55, wherein the second portion of said architecture generates a row precharge instruction to precharge a DRAM row of a bank of the one or more banks of DRAM to cause data to be ready prior to issuing a read command.

58. (Currently amended) The method of Claim 55, wherein the set of architectural registers comprises a register file comprising a plurality of registers and having a parallel access port operative to load or store, under control of the second portion of said program, contents of said register file in a single DRAM access cycle from or to a DRAM row of a bank of the one or more banks of DRAM.

59. (Previously presented) The method of Claim 58, wherein the load operation is performed with a mask to allow certain of the contents of selected registers of the register file not to be modified by the load operation.

60. (Currently amended) The method of Claim 58, wherein said register file further comprises at least a second access port operative to transfer data between one or more of the functional units of the first portion of said architecture.

61. (Currently amended) The method of Claim 58, wherein the first portion and the second portion of said architecture cooperate to execute a DRAM row selected by a row-address register, and said register file further comprises at least a second access port operative to transfer data between one or more of the functional units of the first portion of said architecture.

62. (Currently amended) The method of Claim 58, wherein the register file can be placed into an inactive state where the register file does not appear in ~~the~~ a register space of the functional units of the first portion.

63. (Currently amended) The method of Claim 58, wherein when the register file is placed into ~~the~~ an inactive state, the second portion of said architecture is enabled to cause a parallel load or store operation to occur between the parallel access port and a row of DRAM in a bank of the one or more banks of DRAM.

64. (Previously presented) The method of Claim 55, wherein the first and second portions of said architecture cooperatively execute instructions to process at least one of image data and video data, for display.

65. (Currently amended) The method of Claim 55, wherein the first and second portions of said program cooperatively execute instructions to perform digital filtering operations.

66. (Previously presented) The method of Claim 55, wherein when the first and second portions of said architecture cooperatively execute instructions to execute a video decoder algorithm.

67. (Currently amended) In an embedded-DRAM processor, a method for intelligent caching, comprising:

providing an architecture comprising first and second portions, said first portion comprising a set of functional units and a set of architectural registers exercised thereby, said second portion comprising at least one functional unit capable of moving data between a main memory implemented substantially as one or more banks of DRAM, and said set of architectural registers, wherein the second portion accesses main memory without a caching system that employs cache hits and cache misses; and

providing a program comprising first and second program portions which each concurrently execute subsets of parallely dispatched instructions from one or more instruction streams, said first program portion executed on said first portion of the architecture, said second program portion executed on said second portion of said architecture;

wherein said second portion of said architecture is operative to prefetch data from said main memory into one or more of said architectural registers prior to being processed by said first portion of said architecture, and wherein said second portion of said architecture is operative to move results produced by said first portion of said architecture into main memory after they are produced by said first portion of said architecture; and

prior to when said first portion of said architecture executes a conditional instruction having a condition, said second portion of said architecture prefetches first and second data sets from said main memory into said architectural registers, said first data set being needed for use as instruction operands when said condition is true, said second data set being needed for use as instruction operands when said condition is false.

68. (Previously presented) The method of Claim 67, wherein speculative prefetching of data is performed from said main memory so that the first program portion need not wait for the first or the second data set to be fetched from said main memory, irrespective of the outcome of the conditional instruction.

69. (Previously presented) The method of Claim 67, wherein the second portion generates a row precharge instruction to precharge a DRAM row of a bank of the one or more banks of DRAM to cause data to be ready prior to issuing a read command.

70. (Previously presented) The method of Claim 67, wherein the set of architectural registers comprises a register file comprising a plurality of registers and having a parallel access port operative to load or store, under control of the second portion, contents of said register file in a single DRAM access cycle from or to a DRAM row of a bank of the one or more banks of DRAM.

71. (Previously presented) The method of Claim 70, wherein the load operation is performed with a mask to allow certain of the contents of selected registers of the register file not to be modified by the load operation.

72. (Previously presented) The method of Claim 70, wherein said register file further comprises at least a second access port operative to transfer data between one or more of the functional units of the first portion.

73. (Previously presented) The method of Claim 70, wherein the first portion and the second portion of said architecture cooperate to execute a DRAM row selected by a row-address register, and said register file further comprises at least a second access port operative to transfer data between one or more of the functional units of the first portion.

74. (Currently amended) The method of Claim 70, wherein the register file can be placed into an inactive state where the register file does not appear in ~~the~~ a register space of the functional units of the first portion.

75. (Currently amended) The method of Claim 70, wherein when the register file is placed into ~~the~~ an inactive state, the second portion is enabled to cause a parallel load or store operation to occur between the parallel access port and a row of DRAM in a bank of the one or more banks of DRAM.

76. (Previously presented) The method of Claim 67, wherein the first and second portions of said architecture cooperatively execute instructions to process at least one of image data or video data for display.

77. (Previously presented) The method of Claim 67, wherein the first and second portions of said architecture cooperatively execute instructions to perform digital filtering operations.

78. (Previously presented) The method of Claim 67, wherein when the first and second portions of said architecture cooperatively execute instructions to execute a video decoder algorithm.

79. (Currently amended) In an Embedded-RAM processing apparatus, a method for intelligent caching comprising:

utilizing an architecture comprising first and second portions, said first portion comprising a set of functional units and a set of architectural registers exercised thereby, said second portion comprising at least one functional unit capable of moving data between a main memory implemented as one or more banks of RAM, and said set of architectural registers, wherein the second portion accesses main memory without a caching system that employs cache hits and cache misses; and

utilizing a single program having first and second concurrently executing portions to each concurrently execute distinct subsets of parallely dispatched instructions from one or more instruction streams, said first portion of said program executed on said first portion of the architecture, said second portion of said program executed on said second portion of said architecture;

wherein said second portion of said architecture is operative to fetch data in said main memory prior to being loaded into said first portion of said architecture, and wherein said second portion of said architecture is operative to move results produced by said first portion of said architecture into main memory after they are produced by said first portion of said architecture; and

prior to when said first portion of said architecture executes a conditional instruction <u>having a condition</u>, said second portion of said architecture precharges first and second data sets in respective RAM rows, said first data set being needed for use as instruction operands when said condition is true, said second data set being needed for use as instruction operands when said condition is false.

80. (Previously presented) The method of Claim 79, wherein the second portion of said architecture generates a row precharge instruction to precharge a RAM row of a bank of the one or more banks of RAM to cause data to be ready prior to issuing a read command.

81. (Currently amended) The method of Claim 79, wherein the set of architectural registers comprises a register file comprising a plurality of registers and having a parallel access port operative to load or store, under control of the second portion of said program, contents of said register file in a single RAM access cycle from or to a RAM row of a bank of the one or more banks of RAM.

82. (Previously presented) The method of Claim 81, wherein at least one load operation is performed with a mask to allow certain of the contents of selected registers of the register file not to be modified by the at least one load operation.

83. (Previously presented) The method of Claim 81, wherein said register file further comprises at least a second access port operative to transfer data between one or more of the functional units of the first portion.

84. (Previously presented) The method of Claim 81, wherein the first portion and the second portion of said architecture cooperate to execute a RAM row selected by a row-address register, and said register file further comprises at least a second access port operative to transfer data between one or more of the functional units of the first portion.

85. (Currently amended) The method of Claim 81, wherein the register file can be placed into an inactive state where the register file does not appear in ~~the~~ a register space of the functional units of the first portion.

86. (Currently amended) The method of Claim 81, wherein when the register file is placed into ~~the~~ an inactive state, the second portion of said architecture is enabled to cause a parallel load or store operation to occur between the parallel access port and a row of RAM in a bank of the one or more banks of RAM.

87. (Previously presented) The method of Claim 79, wherein the first and second portions of said architecture cooperatively execute instructions to process at least one of image data or video data for display.

88. (Previously presented) The method of Claim 79, wherein the first and second portions of said architecture cooperatively execute instructions to perform digital filtering operations.

89. (Previously presented) The method of Claim 79, wherein when the first and second portions of said architecture cooperatively execute instructions to execute a video decoder algorithm.

90. (Previously presented) The method of Claim 79, wherein said RAM comprises scroll-RAM.

91. (Previously presented) The method of Claim 79, wherein said RAM comprises synchronous DRAM (SDRAM).

92. (Currently amended) In an Embedded-RAM processing apparatus, a method for intelligent caching comprising:

providing an architecture comprising first and second portions, said first portion comprising a set of functional units and a set of architectural registers exercised thereby, said second portion comprising at least one functional unit capable of moving data between a main memory implemented as one or more banks of RAM, and said set of architectural registers, wherein the second portion accesses main memory without a caching system that employs cache hits and cache misses; and

utilizing a program comprising first and second program portions to each concurrently execute distinct subsets of parallely dispatched instructions from one or more instruction streams, said first program portion executed on said first portion of the architecture, said second program portion executed on said second portion of said architecture;

wherein said second portion of said architecture is operative to prefetch data from said main memory and to pass it into one or more of said architectural registers prior to being processed by said first portion of said architecture, and wherein said second portion of said architecture is operative to move results produced by said first portion of said architecture into main memory after they are produced by said first portion of said architecture; and

prior to when said first portion of said architecture executes a conditional instruction having a condition, said second portion of said architecture prefetches first and second data sets from said main memory into said architectural registers, said first data set being needed for use as instruction operands when said condition is true, said second data set being needed for use as instruction operands when said condition is false.

93. (Previously presented) The method of Claim 92, wherein the second portion of said architecture generates a row precharge instruction to precharge a RAM row of a bank of the one or more banks of RAM to cause data to be ready prior to issuing a read command.

94. (Previously presented) The method of Claim 92, wherein the set of architectural registers comprises a register file comprising a plurality of registers and having a parallel access port operative to load or store, under control of the second portion, contents of said register file in a single RAM access cycle from or to a RAM row of a bank of the one or more banks of RAM.

95. (Previously presented) The method of Claim 94, wherein at least one load operation is performed with a mask to allow certain of the contents of selected registers of the register file not to be modified by the at least one load operation.

96. (Previously presented) The method of Claim 94, wherein said register file further comprises at least a second access port operative to transfer data between one or more of the functional units of the first portion.

97. (Previously presented) The method of Claim 94, wherein the first portion and the second portion of said architecture cooperate to execute a RAM row selected by a row-address register, and said register file further comprises at least a second access port operative to transfer data between one or more of the functional units of the first portion.

98. (Currently amended) The method of Claim 94, wherein the register file can be placed into an inactive state where the register file does not appear in ~~the~~ a register space of the functional units of the first portion.

99. (Currently amended) The method of Claim 94, wherein when the register file is placed into ~~the~~ an inactive state, the second portion is enabled to cause a parallel load or store operation to occur between the parallel access port and a row of RAM in a bank of the one or more banks of RAM.

100. (Previously presented) The method of Claim 92, wherein the first and second portions of said architecture cooperatively execute instructions to process at least one of image data or video data for display.

101. (Previously presented) The method of Claim 92, wherein the first and second portions of said architecture cooperatively execute instructions to perform digital filtering operations.

102. (Previously presented) The method of Claim 92, wherein when the first and second portions of said architecture cooperatively execute instructions to execute a video decoder algorithm.

103. (Previously presented) The method of Claim 92, wherein said RAM comprises scroll-RAM.

104. (Previously presented) The method of Claim 92, wherein said RAM comprises synchronous DRAM (SDRAM).

105. (Currently amended) In an Embedded-DRAM processor, a method for caching comprising:

providing an architecture comprising first and second portions, said first portion comprising a set of functional units and a set of <u>architectural</u> registers exercised thereby, said second portion comprising at least one functional unit capable of moving data between a main memory implemented as one or more banks of DRAM, and said set of architectural registers, the second portion being capable of accessing main memory without a caching system that employs cache hits or cache misses; and

utilizing a program comprising first and second program portions which each concurrently execute distinct subsets of parallely dispatched instructions from one or more instruction streams, said first program portion executed on said first portion of the architecture, said second program portion executed on said second portion of said architecture;

wherein said second portion of said architecture is operative to prefetch data from said main memory and to pass it to one or more of said registers prior to being processed by said first portion of said architecture, and wherein said second portion of said architecture is operative to move results produced by said first portion of said architecture to frame buffer memory after they are produced by said first portion of said architecture; and

prior to when said first portion of said architecture executes a conditional instruction <u>having a condition</u>, said second portion of said architecture prefetches first and second data sets from main memory into said registers, said first data set being needed for use as instruction operands when said condition is true, said second data set being needed for use as instruction operands when said condition is false.

106. (Previously presented) The method of Claim 105, wherein the second portion of said architecture generates a row precharge instruction to precharge a DRAM row of a bank of the one or more banks of DRAM to cause data to be ready prior to issuing a read command.

107. (Currently amended) The method of Claim 105, wherein the set of architectural registers comprises a register file comprising a plurality of registers and having a parallel access port operative to load or store, under control of the second portion of said program, contents of said register file in a single DRAM access cycle from or to a DRAM row of a bank of the one or more banks of DRAM.

108. (Previously presented) The method of Claim 107, wherein at least one load operation is performed with a mask to allow certain of the contents of selected registers of the register file not to be modified by the at least one load operation.

109. (Previously presented) The method of Claim 107, wherein said register file further comprises at least a second access port operative to transfer data between one or more of the functional units of the first portion.

110. (Previously presented) The method of Claim 107, wherein the first portion and the second portion of said architecture cooperate to execute a RAM row selected by a row-address register, and said register file further comprises at least a second access port operative to transfer data between one or more of the functional units of the first portion.

111. (Currently amended) The method of Claim 107, wherein the register file can be placed into an inactive state where the register file does not appear in ~~the~~ a register space of the functional units of the first portion.

112. (Currently amended) The method of Claim 107, wherein when the register file is placed into ~~the~~ an inactive state, the second portion is enabled to cause a parallel load or store operation to occur between the parallel access port and a row of RAM in a bank of the one or more banks of RAM.

113. (Previously presented) The method of Claim 105, wherein the first and second portions of said architecture cooperatively execute instructions to process at least one of image data or video data for display.

114. (Previously presented) The method of Claim 105, wherein the first and second portions of said architecture cooperatively execute instructions to perform digital filtering operations.

115. (Previously presented) The method of Claim 105, wherein when the first and second portions of said architecture cooperatively execute instructions to execute a video decoder algorithm.

116. (Previously presented) For use in a system involving an Embedded-DRAM processor, a method for caching comprising:

providing an architecture comprising first and second portions, said first portion comprising a set of functional units and a set of architectural registers exercised thereby, said second portion comprising at least one functional unit capable of moving data between a main memory implemented as one or more banks of DRAM, and said set of architectural registers, the second portion accessing main memory without a caching system that employs cache hits and cache misses; and

providing a program comprising first and second program portions concurrently executing respective ones of distinct subsets of parallely dispatched instructions from one or more instruction streams, said first program portion executing on said first portion of the architecture, said second program portion executing on said second portion of said architecture;

wherein said second portion of said architecture is operative to prefetch data from said main memory and to pass it into one or more of said architectural registers prior to processing by said first portion of said architecture, and wherein said second portion of said architecture is operative to move results produced by said first portion of said architecture into a frame buffer memory after they are produced by said first portion of said architecture; and

prior to when said first portion of said architecture executes an instruction, said second portion of said architecture prefetching first and second data sets from main memory into said architectural registers, said first and second data sets being needed for use as instruction operands.

117. (Previously presented) The method of Claim 116, wherein the second portion of said architecture generates a row precharge instruction to precharge a DRAM row of a bank of the one or more banks of DRAM to cause data to be ready prior to issuing a read command.

118. (Previously presented) The method of Claim 116, wherein the set of architectural registers comprises a register file comprising a plurality of registers and having a parallel access port operative to load or store, under control of the second portion, contents of said register file in a single DRAM access cycle from or to a DRAM row of a bank of the one or more banks of DRAM.

119. (Previously presented) The method of Claim 118, wherein at least one load operation is performed with a mask to allow certain of the contents of selected registers of the register file not to be modified by the at least one load operation.

120. (Previously presented) The method of Claim 118, wherein said register file further comprises at least a second access port operative to transfer data between one or more of the functional units of the first portion.

121. (Currently amended) The method of Claim 118, wherein ~~the first portion and the second portion of said architecture cooperate to execute a RAM row selected by a row-address register, and~~ said register file further comprises at least a second access port operative to transfer data between one or more of the functional units of the first portion of said architecture.

122. (Currently amended) The method of Claim 118, wherein the register file can be placed into an inactive state where the register file does not appear in ~~the~~ a register space of the functional units of the first portion.

123. (Currently amended) The method of Claim 118, wherein when the register file is placed into ~~the~~ an inactive state, the second portion is enabled to cause a parallel load or store operation to occur between the parallel access port and a row of RAM in a bank of the one or more banks of RAM.

124. (Previously presented) The method of Claim 116, wherein the first and second portions of said architecture cooperatively execute instructions to process at least one of image data or video data for display.

125. (Previously presented) The method of Claim 116, wherein the first and second portions of said architecture cooperatively execute instructions to perform digital filtering operations.

126. (Previously presented) The method of Claim 116, wherein when the first and second portions of said architecture cooperatively execute instructions to execute a video decoder algorithm.

127. (Currently amended) For use in a system involving an Embedded-DRAM processor, said system comprising (i) an architecture comprising first and second portions, said first portion comprising a set of functional units and a set of architectural registers exercised thereby, said second portion comprising at least one functional unit capable of moving data between a main memory implemented as one or more banks of DRAM, and said set of architectural registers, wherein the second portion accesses main memory without a caching system that employs cache hits and cache misses, and (ii) a program comprising first and second program portions which each concurrently execute distinct subsets of parallely dispatched instructions from one or more instruction streams, said first program portion executed on said first portion of the architecture, said second program portion executed on said second portion of said architecture, a method for intelligent caching comprising:

prefetching, using at least said second portion of said architecture, data from said main memory into one or more of said architectural registers prior to being processed by said first portion of said architecture, and wherein said second portion of said architecture is operative to move results produced by said first portion of said architecture into main memory after they are produced by said first portion of said architecture; and

prefetching, using at least said second portion of said architecture and prior to when said first portion of said architecture executes a conditional branch instruction <u>having a condition,</u> first and second data sets from said main memory into said architectural registers, said first data set being needed for use as instruction operands when said condition is true, said second data set being needed for use as instruction operands when said condition is false.

128. (Currently amended) For use in a system involving an Embedded-RAM processor, said system comprising (i) an architecture comprising first and second portions, said first portion comprising a set of functional units and a set of architectural registers exercised thereby, said second portion comprising at least one functional unit capable of moving data

between a main memory implemented as one or more banks of RAM, and said set of architectural registers, wherein the second portion accesses main memory without a caching system that employs cache hits and cache misses, and (ii) a program comprising first and second program portions which each concurrently execute distinct subsets of parallely dispatched instructions from one or more instruction streams, said first program portion executed on said first portion of the architecture, said second program portion executed on said second portion of said architecture, a method for intelligent caching comprising:

fetching, using at least said second portion of said architecture, data in said main memory prior to being loaded into said first portion of said architecture, and wherein said second portion of said architecture is operative to move results produced by said first portion of said architecture into main memory after they are produced by said first portion of said architecture; and

precharging, using at least said second portion of said architecture, and prior to when said first portion of said architecture executes a conditional instruction having a condition, first and second data sets in respective RAM rows, said first data set being needed for use as instruction operands when said condition is true, said second data set being needed for use as instruction operands when said condition is false.

129. (Currently amended) In an Embedded-RAM processing apparatus comprising (i) an architecture comprising first and second portions, said first portion comprising a set of functional units and a set of architectural registers exercised thereby, said second portion comprising at least one functional unit capable of moving ~~data between a~~ at least a portion of the contents of a main memory implemented as one or more banks of RAM~~, and~~ to said set of architectural registers, wherein the second portion accesses main memory without a caching system that employs cache hits and cache misses, and (ii) a program comprising first and second program portions which concurrently execute distinct subsets of parallely dispatched instructions from one or more instruction streams, said first program portion executed on said first portion of the architecture, said second program portion executed on said second portion of said architecture, a method for intelligent caching comprising:

fetching, using at least said second portion of said architecture, data in said main memory prior to said data being passed to one or more of said architectural registers prior to said data being processed by said first portion of said architecture, and wherein said second portion of said architecture is operative to move results produced by said first portion of said architecture into said main memory after they are produced by said first portion of said architecture; and

precharging, using at least said second portion of said architecture, and prior to when said first portion of said architecture executes a conditional instruction having a condition, from said main memory into said architectural registers, first and second data sets, said first data set being needed for use as instruction operands when said condition is true, said second data set being needed for use as instruction operands when said condition is false.

130. (Currently amended) In an Embedded-DRAM processing apparatus comprising (i) an architecture split into first and second portions, said first portion comprising a set of functional units and a set of architectural registers exercised thereby, said second portion comprising at least one functional unit capable of moving ~~data between a~~ at least a portion of the contents of a main memory implemented as one or more banks of DRAM~~, and~~ to said set of architectural registers, wherein the second portion accesses main memory without a caching system that employs cache hits and cache misses, and (ii) a program comprising first and second program portions which concurrently execute distinct subsets of parallely dispatched instructions from one or more instruction streams, said first program portion executed on said first portion of the architecture, said second program portion executed on said second portion of said architecture, a method for intelligent caching comprising:

fetching, using at least said second portion of said architecture, data in said main memory prior to said data being passed to one or more of said architectural registers prior to said data being processed by said first portion of said architecture, and wherein said second portion of said architecture is operative to move results produced by said first portion of said architecture into a frame buffer memory after they are produced by said first portion of said architecture; and

charging, using at least said second portion of said architecture, and prior to when said first portion of said architecture executes a conditional instruction, from said main memory into said architectural registers, first and second data sets, said first and second data sets being needed for use as instruction operands.

131. (Currently amended) In an Embedded-DRAM processing apparatus comprising (i) an architecture comprising first and second portions, said first portion comprising a set of functional units and a set of architectural registers exercised thereby, said second portion comprising at least one functional unit capable of moving ~~data between a~~ at least a portion of the

contents of a main memory implemented as one or more banks of DRAM~~, and~~ to said set of architectural registers, wherein the second portion accesses main memory without a caching system that employs cache hits and cache misses, and (ii) a program comprising first and second program portions which concurrently execute distinct subsets of parallely dispatched instructions from one or more instruction streams, said first program portion executed on said first portion of the architecture, said second program portion executed on said second portion of said architecture, a method for intelligent caching comprising:

fetching, using at least said second portion of said architecture, data in said main memory prior to said data being passed to one or more of said architectural registers prior to said data being processed by said first portion of said architecture, and wherein said second portion of said architecture is operative to move results produced by said first portion of said architecture into said main memory after they are produced by said first portion of said architecture; and

charging, using at least said second portion of said architecture, and prior to when said first portion of said architecture executes a conditional instruction having a condition, in respective DRAM rows, first and second data sets, said first data set being needed for use as instruction operands when said condition is true, said second data set being needed for use as instruction operands when said condition is false.

132. (Currently amended) In an Embedded-DRAM processing apparatus comprising (i) an architecture comprising first and second portions, said first portion comprising a set of functional units and a set of architectural registers exercised thereby, said second portion comprising at least one functional unit capable of moving ~~data between a~~ at least a portion of the contents of a main memory implemented as one or more banks of DRAM~~, and~~ to said set of registers, wherein the second portion accesses main memory without a caching system that employs cache hits and cache misses, and (ii) a program comprising first and second program portions which concurrently execute distinct subsets of parallely dispatched instructions from one or more instruction streams, said first program portion executed on said first portion of the architecture, said second program portion executed on said second portion of said architecture, a method for intelligent caching comprising:

fetching, using at least said second portion of said architecture, data in said main memory prior to said data being passed to one or more of said registers prior to said data being processed by said first portion of said architecture, and wherein said second portion of said architecture is operative to move results produced by said first portion of said architecture into said main memory after they are produced by said first portion of said architecture; and

prefetching, using at least said second portion of said architecture, and prior to when said first portion of said architecture executes a conditional instruction, in respective DRAM rows, first and second data sets into said registers, said first and second data sets being needed for use as operands.

133. (Currently amended) For use in an Embedded-DRAM processing apparatus (i) first architecture means and second architecture means, said first means comprising a set of functional means and a set of architectural register means exercised thereby, said second architecture means comprising at least one functional means capable of moving ~~data between~~ at least a portion of the contents of one or more banks of means for storing data~~, and~~ to said set of register means, wherein the second architectural means accesses said means for storing data without a caching system that employs cache hits and cache misses, and (ii) a program comprising first and second program portions which each concurrently execute distinct subsets of parallely dispatched instructions from one or more instruction streams, said first program portion executed on said first architecture means, said second program portion executed on said second architecture means, a method for caching comprising:

prefetching, using at least said second architecture means, data from said means for storing data into one or more of said register means prior to said data being processed by said first architecture means, and wherein said second architecture means is operative to move results produced by said first architecture means into said means for storing after they are produced by said first architecture means; and

prefetching, using at least said second architecture means and prior to when said first architecture means executes a conditional instruction having a condition, first and second data sets from said means for storing into said register means, said first data set being needed for use as operands when said condition is true, said second data set being needed for use as operands when said condition is false.

134. (Currently amended) In an Embedded-RAM processing apparatus comprising (i) an architecture comprising first and second portions, said first portion comprising a set of functional means and a set of registers exercised thereby, said second portion comprising at least one functional means capable of at least moving ~~data between~~ contents of a main memory

implemented as one or more banks of RAM~~, and~~ to said set of registers, wherein the second portion accesses main memory without a caching system that employs cache hits and cache misses, and (ii) a program comprising first and second program portions which concurrently execute distinct subsets of parallely dispatched instructions from one or more instruction streams, said first program portion executed on said first portion of the architecture, said second program portion executed on said second portion of said architecture, a method for caching comprising:

fetching, using at least said second portion of said architecture, data in said main memory prior to said data being passed to one or more of said registers prior to said data being processed by said first portion of said architecture, and wherein said second portion of said architecture is operative to move results produced by said first portion of said architecture into said main memory after they are produced by said first portion of said architecture; and

precharging first and second data sets into said registers, using at least said second portion of said architecture, and prior to when said first portion of said architecture executes a conditional instruction having a condition, said first data set being needed for use as operands when said condition is true, said second data set being needed for use as operands when said condition is false.

135. (Currently amended) In an Embedded-RAM processing apparatus comprising (i) an architecture comprising first and second portions, said first portion comprising a set of functional means and a set of registers exercised thereby, said second portion comprising at least one functional means capable of at least moving ~~data between~~ contents of a main memory implemented as one or more banks of RAM~~, and~~ to said set of registers, wherein the second portion accesses main memory without a caching system that employs cache hits and cache misses, and (ii) a program comprising first and second program portions which concurrently execute distinct subsets of parallely dispatched instructions from one or more instruction streams, said first program portion executed on said first portion of the architecture, said second program portion executed on said second portion of said architecture, a method for caching comprising:

fetching, using at least said second portion of said architecture, data in said main memory prior to said data being passed to one or more of said registers prior to said data being processed by said first portion of said architecture, and wherein said second portion of said architecture is operative to move results produced by said first portion of said architecture into a frame buffer after they are produced by said first portion of said architecture; and charging first and second data sets into said registers, using at least said second portion of said architecture, and prior to when said first portion of said architecture executes a conditional instruction, said first and second data sets being needed for use as operands.